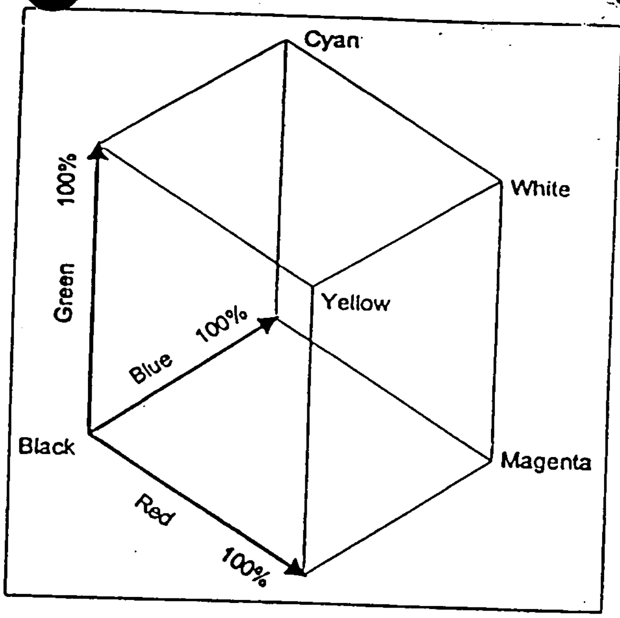


FIG. 1 PRIOR ART

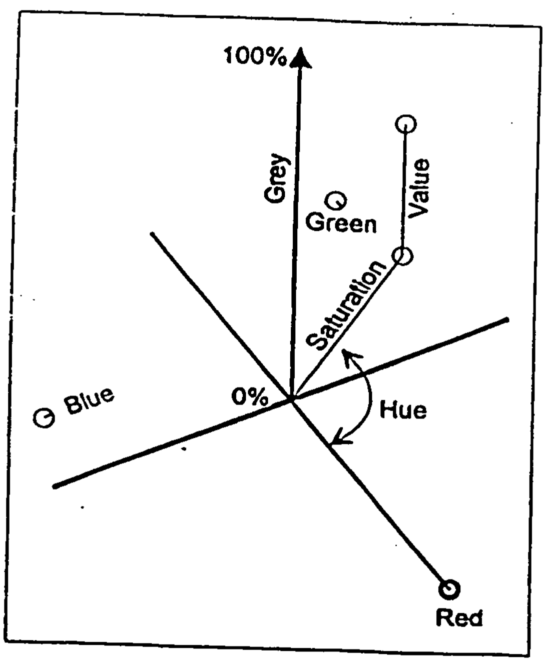


FIG. 2 PRIOR ART

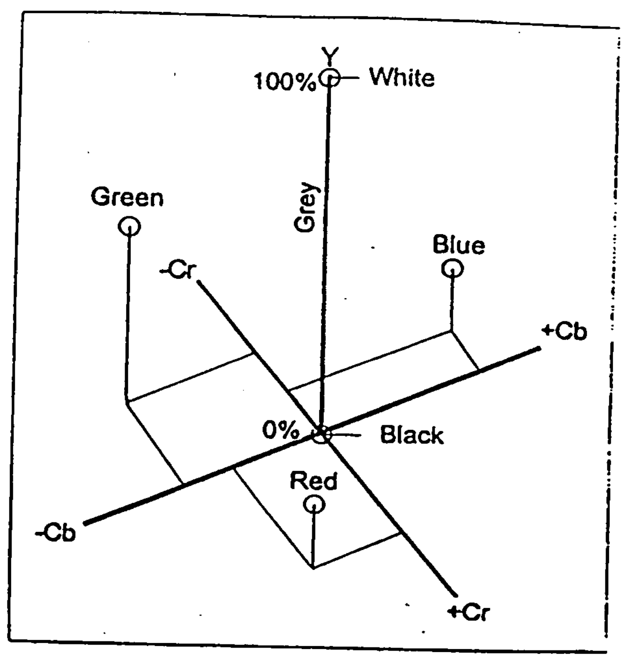


FIG. 3 PRIOR ART

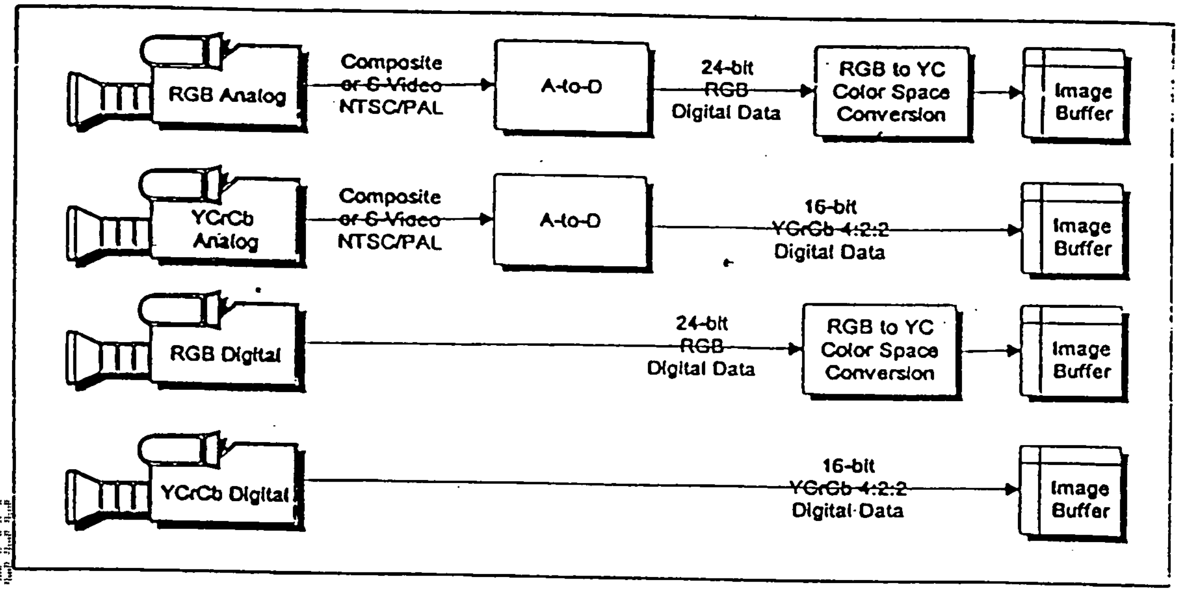

RGB Analog
Composite
~~or S-Video~~
NTSC/PAL
A-to-D
24-bit
~~RGB~~
Digital Data
RGB to YC
Color Space
Conversion
Image
Buffer
YCrCb
Analog
Composite
~~or S-Video~~
NTSC/PAL
A-to-D
16-bit
~~YCrCb 4:2:2~~
Digital Data
Image
Buffer
RGB Digital
24-bit
~~RGB~~
Digital Data
RGB to YC
Color Space
Conversion
Image
Buffer
YCrCb Digital
16-bit
~~YCrCb 4:2:2~~
Digital Data
Image
Buffer

FIG. 4

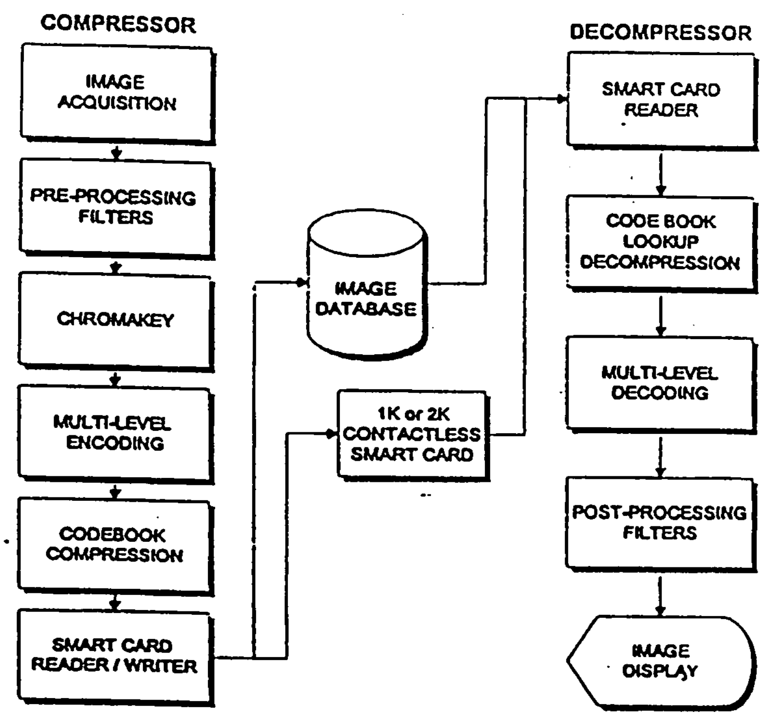

COMPRESSOR
IMAGE
ACQUISITION
PRE-PROCESSING
FILTERS
CHROMAKEY
MULTI-LEVEL
ENCODING
CODEBOOK
COMPRESSION
SMART CARD
READER / WRITER
IMAGE
DATABASE
1K or 2K
CONTACTLESS
SMART CARD
DECOMPRESSOR
SMART CARD
READER
CODE BOOK
LOOKUP
DECOMPRESSION
MULTI-LEVEL
DECODING
POST-PROCESSING
FILTERS
IMAGE
DISPLAY

FIG. 5

If all pixels are within a specified threshold, the output is the average of the four pixels, two on each side of the target.

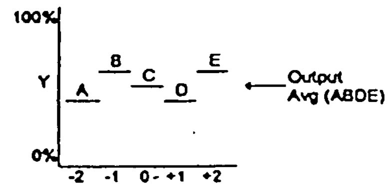


FIG. 6

If the two pixels on either side are within a specified threshold and both sides themselves are within the threshold, the target pixel is considered to be impulse noise. The output is the average of the two pixels on each side of the target.

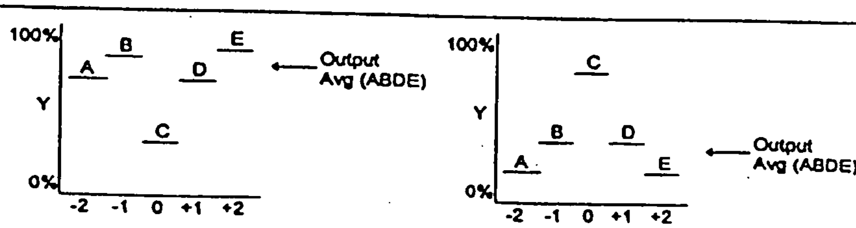
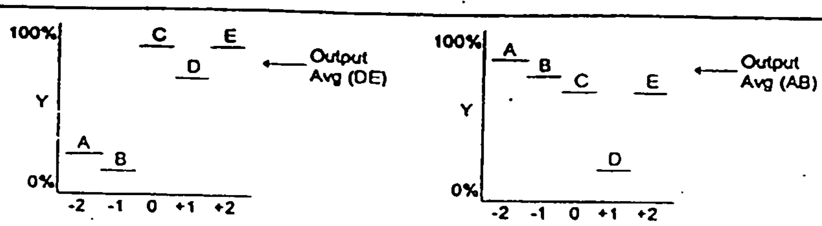


FIG. 7

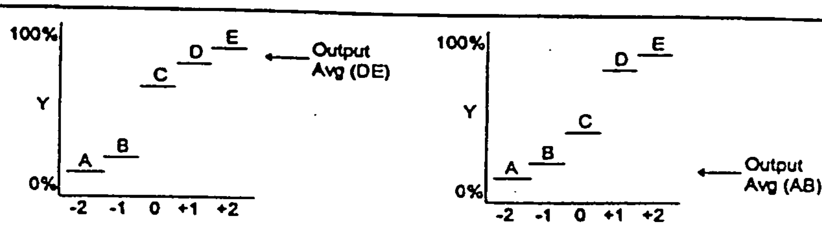
If the two pixels on either side of the target pixel and the target pixel itself are within a specified threshold, the target pixel is



considered to be an edge pixel. The output is the average of the two pixels on the matching side.

FIG. 8

If the five pixels are all increasing or decreasing (or are within a small threshold to account for ringing or pre-emphasis



typically found in analog video signals), the target is considered to be in the midst of a blurred edge. The output is the average of the two pixels on whichever side is closest to the target pixel.

FIG. 9

If the five pixels in the window do not fit into any of the prior cases, the target is considered to be in the midst of a busy area. The target pixel is output unchanged.

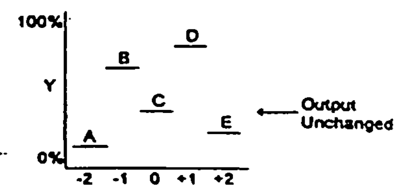


FIG. 10

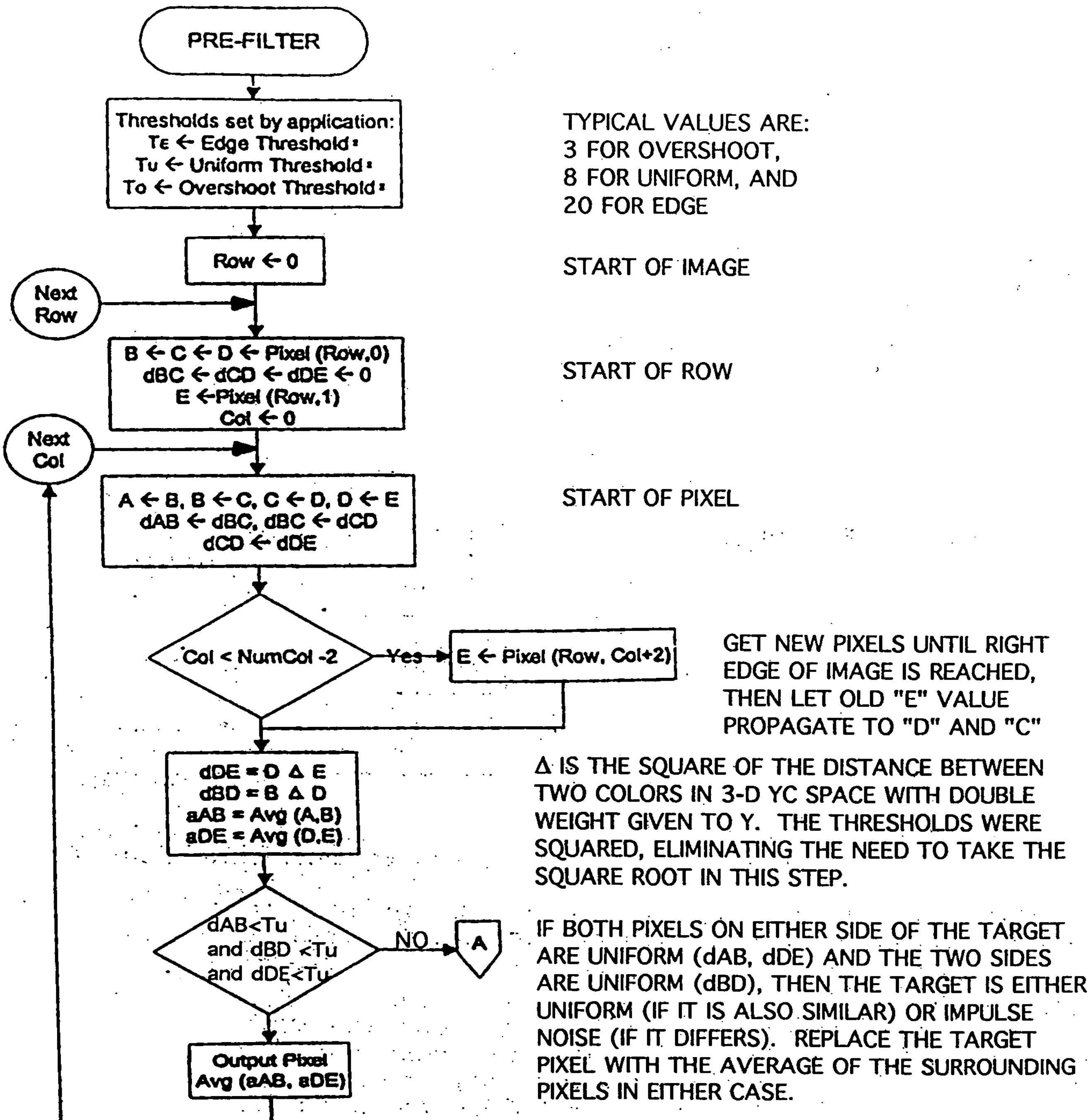
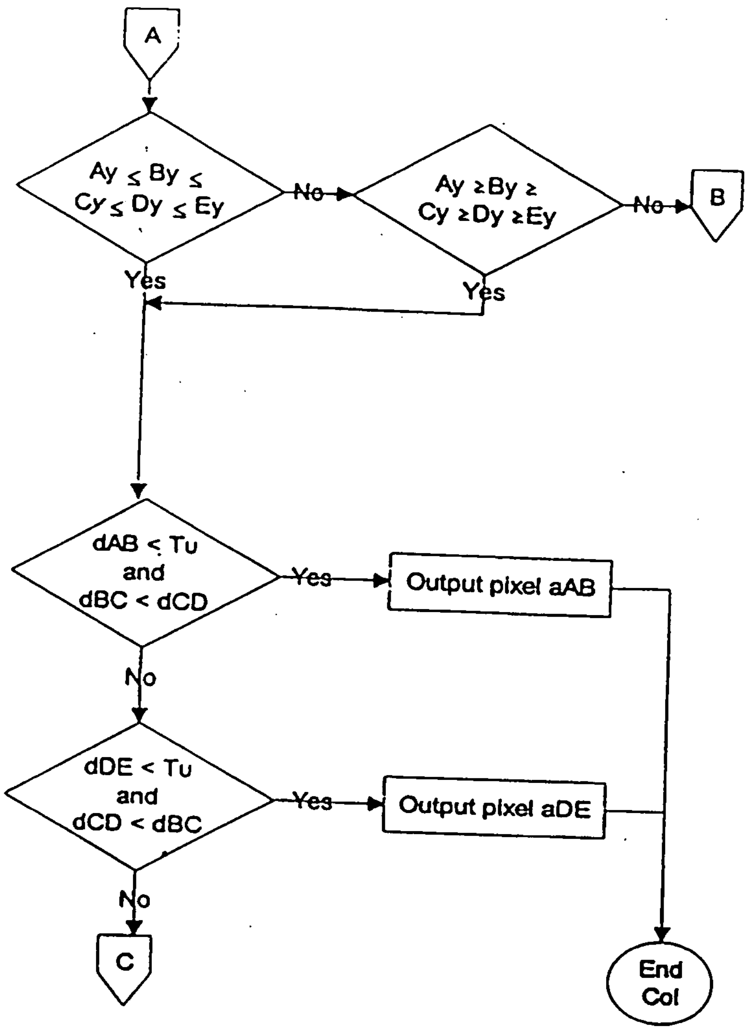

PRE-FILTER
Thresholds set by application:
Tε ← Edge Threshold
Tu ← Uniform Threshold
To ← Overshoot Threshold
TYPICAL VALUES ARE:
3 FOR OVERSHOOT,
8 FOR UNIFORM, AND
20 FOR EDGE
Row ← 0
START OF IMAGE
Next Row
B ← C ← D ← Pixel (Row,0)
dBC ← dCD ← dDE ← 0
E ← Pixel (Row,1)
Col ← 0
START OF ROW
Next Col
A ← B, B ← C, C ← D, D ← E
dAB ← dBC, dBC ← dCD
dCD ← dDE
START OF PIXEL
Col < NumCol -2
Yes
E ← Pixel (Row, Col+2)
GET NEW PIXELS UNTIL RIGHT EDGE OF IMAGE IS REACHED, THEN LET OLD "E" VALUE PROPAGATE TO "D" AND "C"
dDE = D Δ E
dBD = B Δ D
aAB = Avg (A,B)
aDE = Avg (D,E)
Δ IS THE SQUARE OF THE DISTANCE BETWEEN TWO COLORS IN 3-D YC SPACE WITH DOUBLE WEIGHT GIVEN TO Y. THE THRESHOLDS WERE SQUARED, ELIMINATING THE NEED TO TAKE THE SQUARE ROOT IN THIS STEP.
dAB<Tu and dBD <Tu and dDE<Tu
NO
A
IF BOTH PIXELS ON EITHER SIDE OF THE TARGET ARE UNIFORM (dAB, dDE) AND THE TWO SIDES ARE UNIFORM (dBD), THEN THE TARGET IS EITHER UNIFORM (IF IT IS ALSO SIMILAR) OR IMPULSE NOISE (IF IT DIFFERS). REPLACE THE TARGET PIXEL WITH THE AVERAGE OF THE SURROUNDING PIXELS IN EITHER CASE.
Output Pixel
Avg (aAB, aDE)

FIG. 11A



If all pixels are either increasing or decreasing in luminance, a gradient or slow (blurred) edge has been detected.

Note 1.
The overshoot threshold is used to compensate for preemphasis in the analog signal. The comparison is actually:

Ay < By or dAB < To

Note 2.
Comparison by only the luminance components has been shown here. If computational power is available, a more complete solution would be to check that all three components were either decreasing or increasing.

Output the average of the two pixels to whichever side is closest to the target pixel provided those two pixels are sufficiently close to each other so as to not distort the image.

FIG. 11B

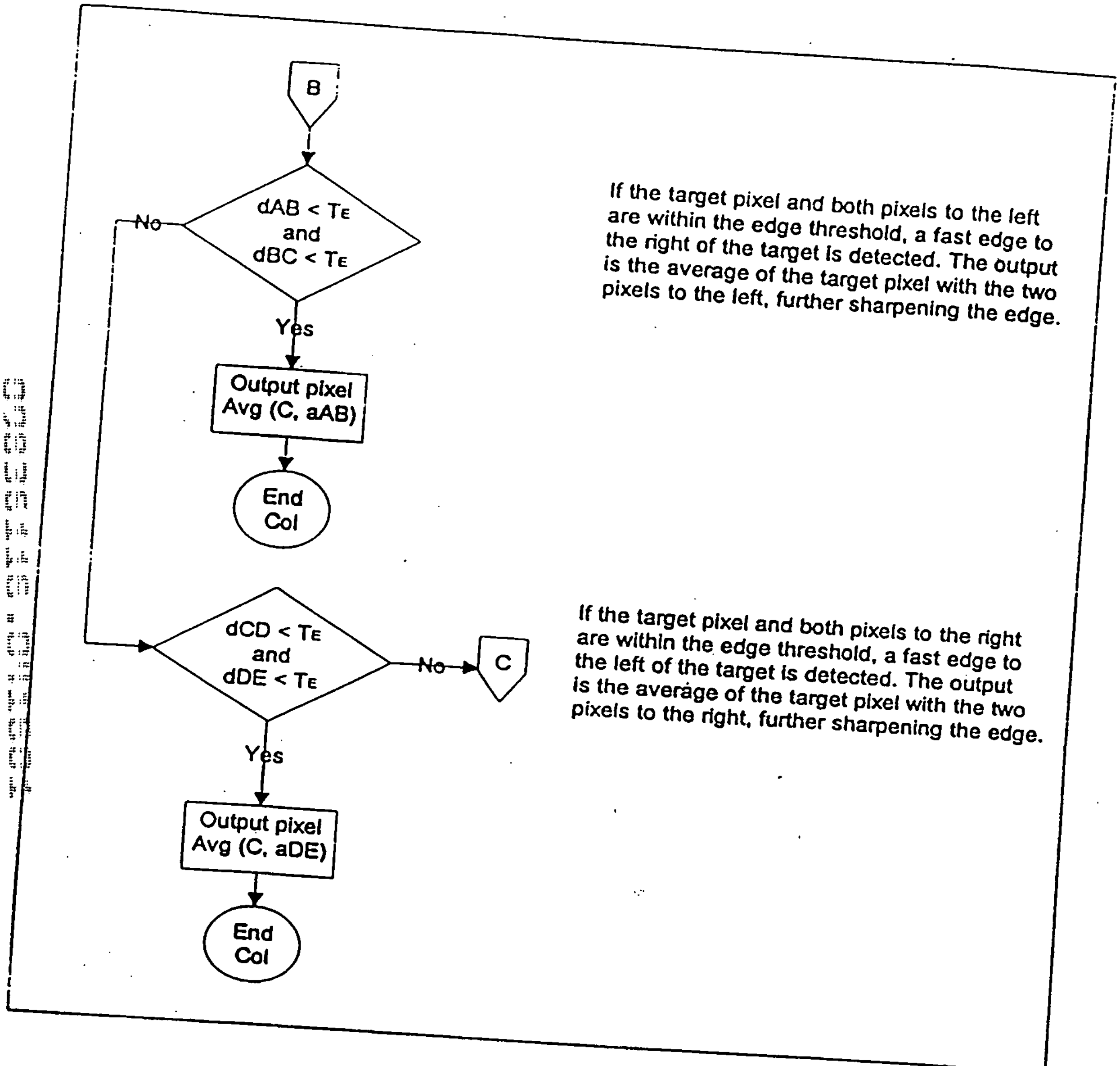
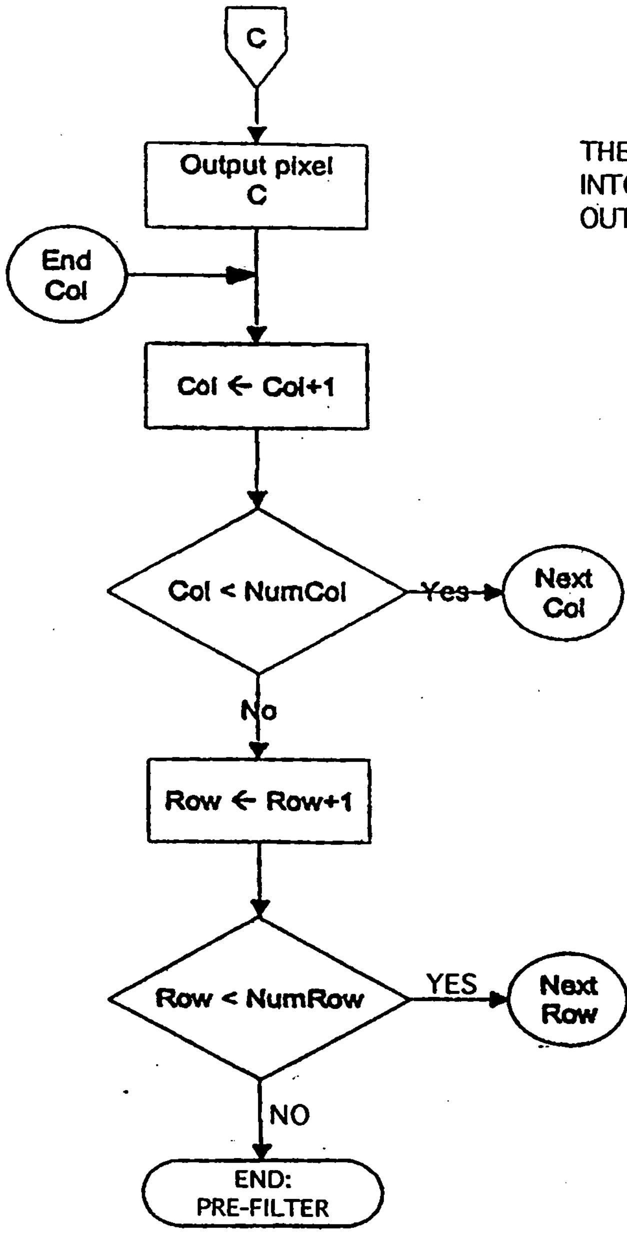

B
dAB < TE
and
dBC < TE
No
Yes
Output pixel
Avg (C, aAB)
End
Col
If the target pixel and both pixels to the left are within the edge threshold, a fast edge to the right of the target is detected. The output is the average of the target pixel with the two pixels to the left, further sharpening the edge.
dCD < TE
and
dDE < TE
No
C
Yes
Output pixel
Avg (C, aDE)
End
Col
If the target pixel and both pixels to the right are within the edge threshold, a fast edge to the left of the target is detected. The output is the average of the target pixel with the two pixels to the right, further sharpening the edge.

FIG. 11C



THE TARGET PIXEL HAS NOT FALLEN INTO ANY OF THE CASES, SO IT IS OUTPUT UNCHANGED.

FIG. 11D

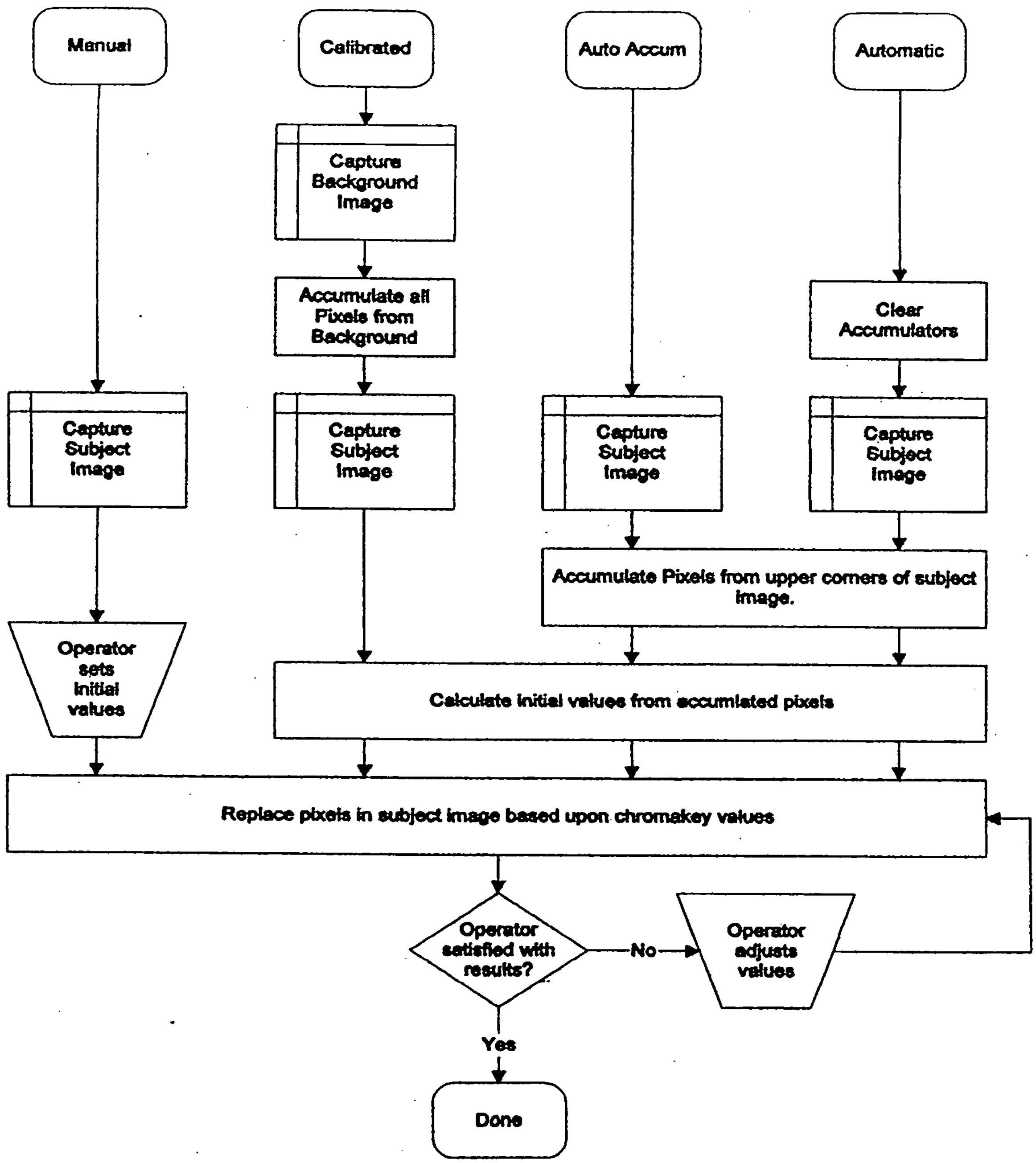

Manual
Calibrated
Auto Accum
Automatic
Capture Background Image
Accumulate all Pixels from Background
Clear Accumulators
Capture Subject Image
Capture Subject Image
Capture Subject Image
Capture Subject Image
Accumulate Pixels from upper corners of subject image.
Operator sets initial values
Calculate initial values from accumlated pixels
Replace pixels in subject image based upon chromakey values
Operator satisfied with results?
No
Operator adjusts values
Yes
Done

FIG. 11E

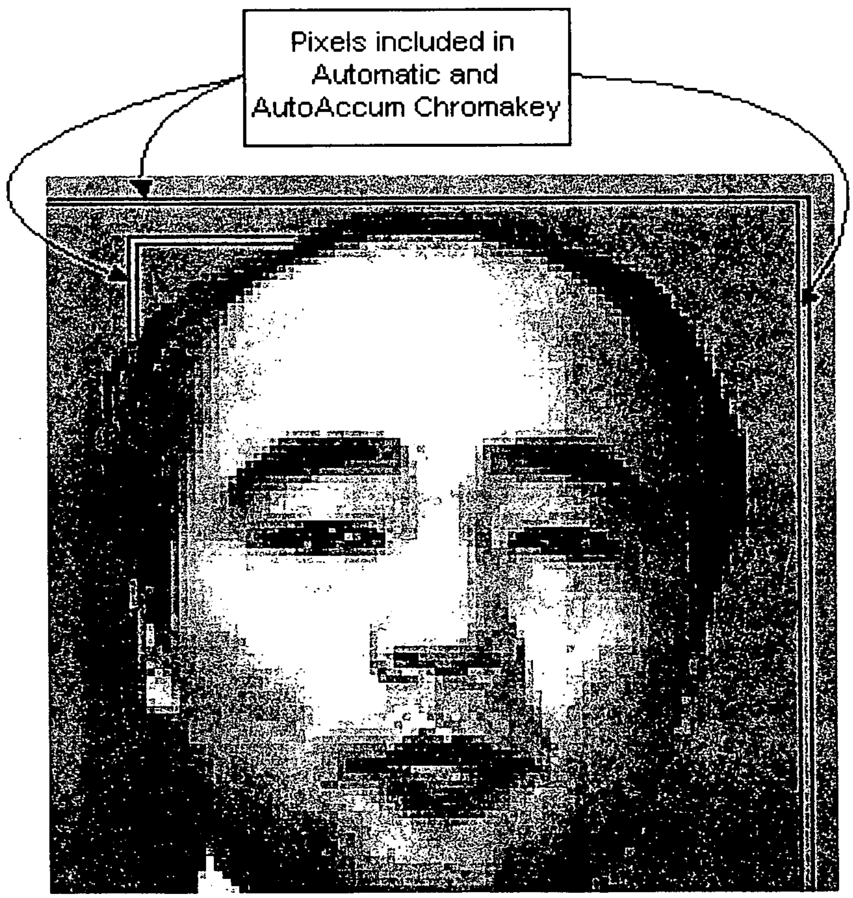
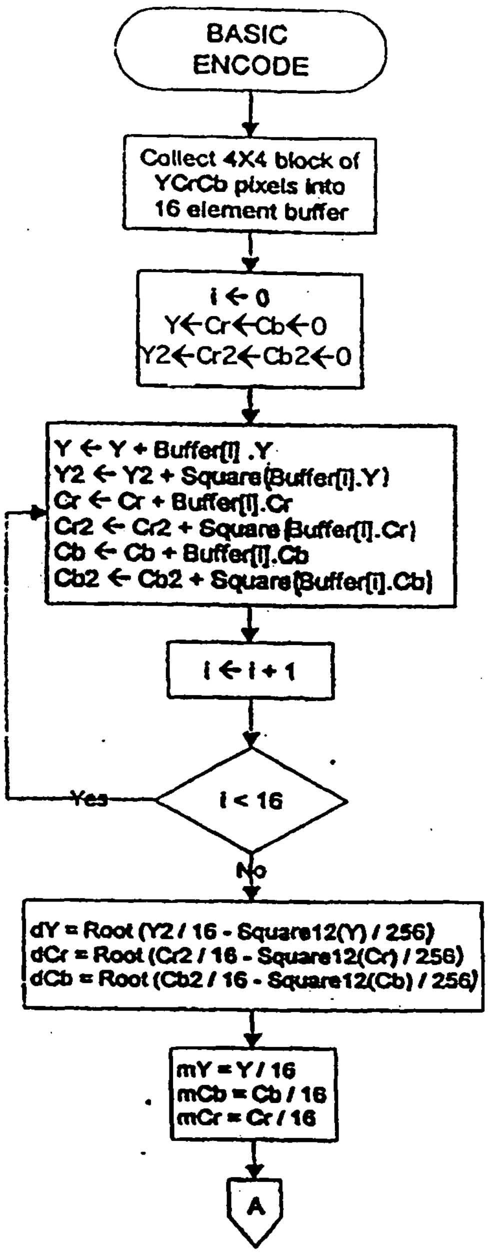


FIG. 11F



Buffer index will range from 0 to 15. Color components will be referred to as: ".Y", ".Cr", and ".Cb"

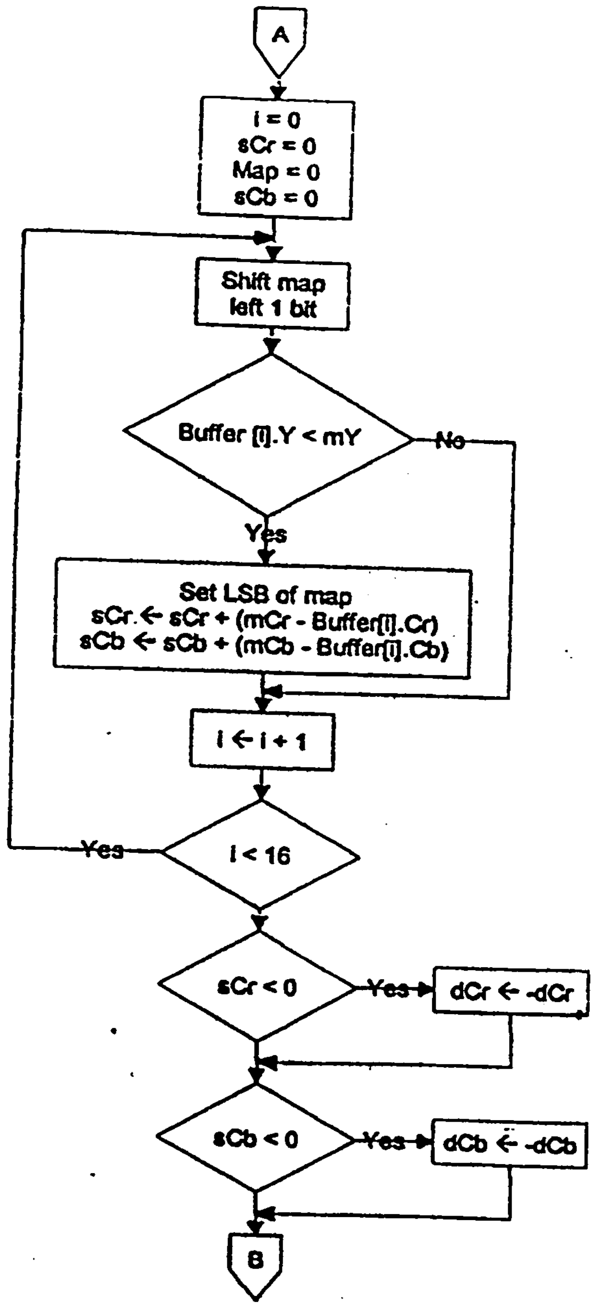
<u>Step 1 - Collect first and second moments</u>

Accumulate separate component values as squares for each pixel. Squares are calculated by table lookup rather than by multiplication.

<u>Step 2 - Calculate mean and standard deviation</u>
The square12 function calculates the square of a 12-bit number using the same 8-bit table of squares above and little extra arithmetic. The root function finds roots by binary search of the 8-bit table of squares.

dY, dCr, and dCb are the standard deviations for each component and mY, mCr, and mCb are the arithmetic means.

FIG. 12A



Step 3 - Determine selector map

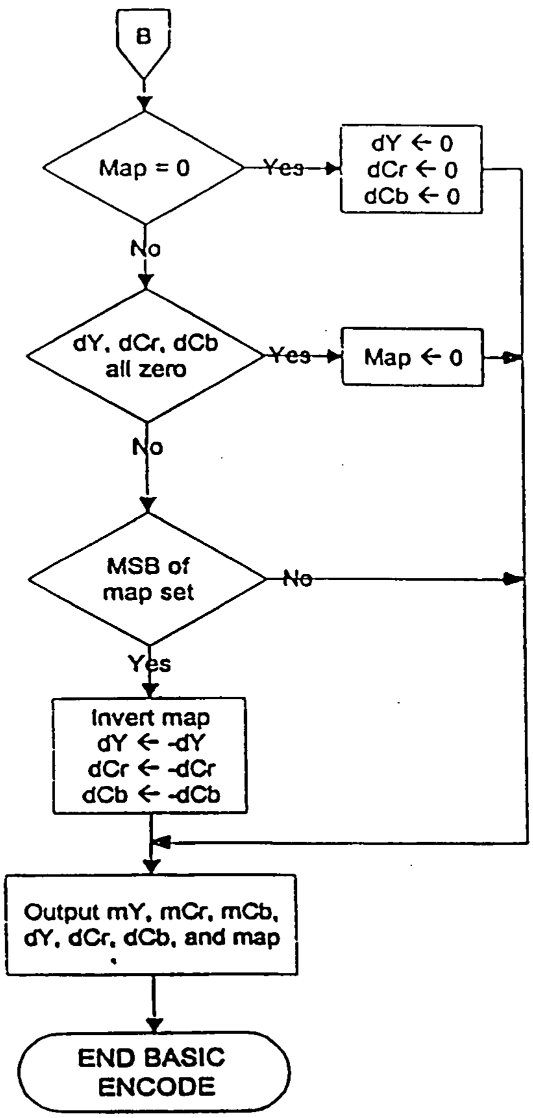
Use the mean luminance value for the selector.

The one bits in the map mark those pixels that are "darker" than the mean. Accumulate the signed differences from the mean in each chromanance channel.

If the Cr channel decreases when the luminance increases, invert dCr.

If the Cb channel decreases when the luminance increases, invert dCb.

FIG. 12B



Step 4 - Normalize

If the luminance of all pixels is equal to (or, in practice, slightly greater than) the mean, zero all standard deviation values.

If all of the pixels are nearly equal, the standard deviations will all be zero. In this case the map is also zeroed.

To reduce the number of possible maps from 65,536 to 32,768, if the map is set, the map is inverted and the dY, dCr, and dCb values are negated.

Typically 8 bits are used for each of the mY, mCr, mCb, dY, dCr, and dCb values and 16 bits are used for the map.

FIG. 12C

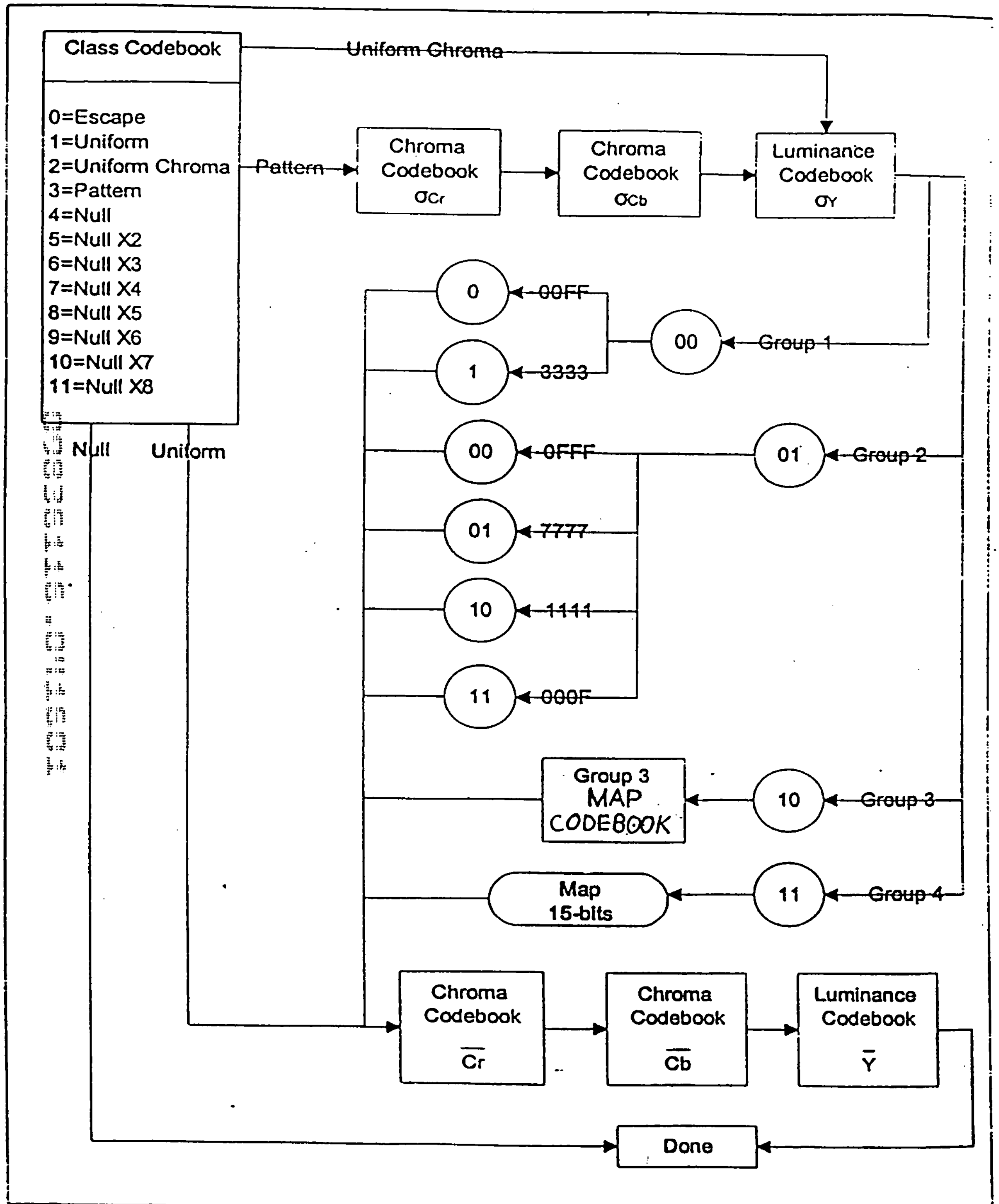

Class Codebook
0=Escape
1=Uniform
2=Uniform Chroma
3=Pattern
4=Null
5=Null X2
6=Null X3
7=Null X4
8=Null X5
9=Null X6
10=Null X7
11=Null X8
Uniform Chroma
Pattern
Chroma Codebook σCr
Chroma Codebook σCb
Luminance Codebook σY
0
00FF
1
3333
00
Group 1
00
0FFF
01
Group 2
01
7777
10
1111
11
000F
Group 3 MAP CODEBOOK
10
Group 3
Map 15-bits
11
Group 4
Null
Uniform
Chroma Codebook Cr
Chroma Codebook Cb
Luminance Codebook Y
Done

FIG. 13

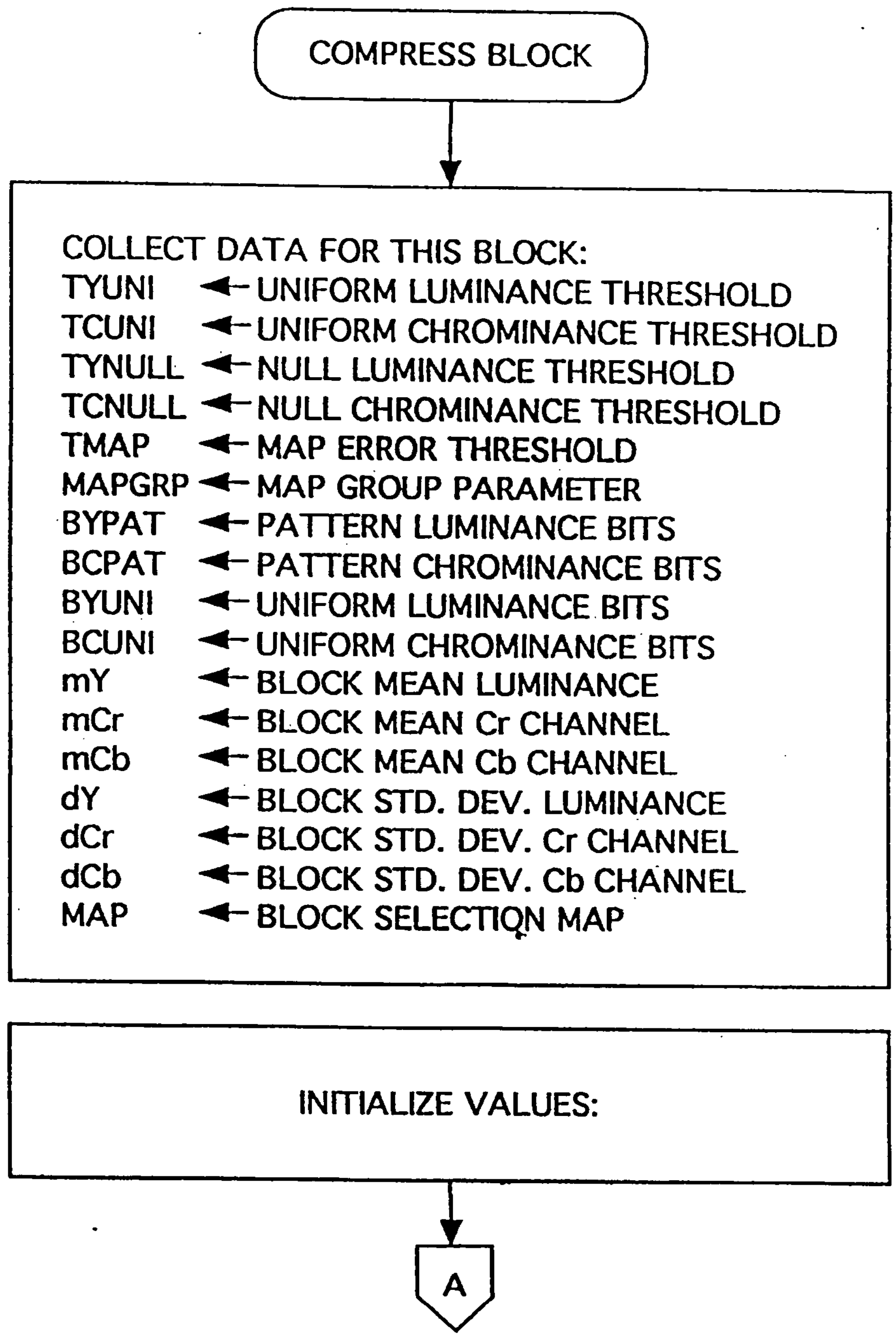

COMPRESS BLOCK
COLLECT DATA FOR THIS BLOCK:
TYUNI ← UNIFORM LUMINANCE THRESHOLD
TCUNI ← UNIFORM CHROMINANCE THRESHOLD
TYNULL ← NULL LUMINANCE THRESHOLD
TCNULL ← NULL CHROMINANCE THRESHOLD
TMAP ← MAP ERROR THRESHOLD
MAPGRP ← MAP GROUP PARAMETER
BYPAT ← PATTERN LUMINANCE BITS
BCPAT ← PATTERN CHROMINANCE BITS
BYUNI ← UNIFORM LUMINANCE BITS
BCUNI ← UNIFORM CHROMINANCE BITS
mY ← BLOCK MEAN LUMINANCE
mCr ← BLOCK MEAN Cr CHANNEL
mCb ← BLOCK MEAN Cb CHANNEL
dY ← BLOCK STD. DEV. LUMINANCE
dCr ← BLOCK STD. DEV. Cr CHANNEL
dCb ← BLOCK STD. DEV. Cb CHANNEL
MAP ← BLOCK SELECTION MAP
INITIALIZE VALUES:
A

FIG. 14A

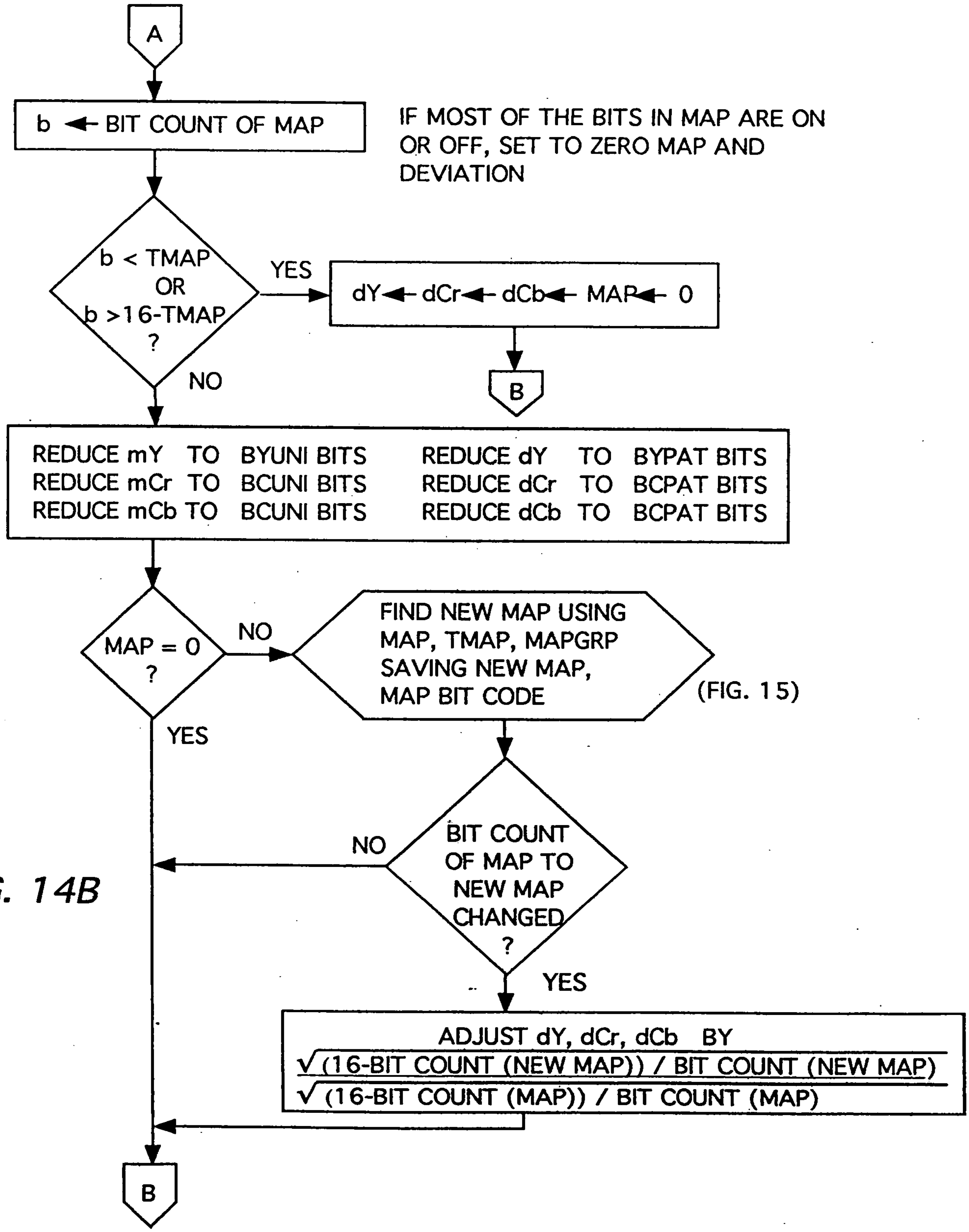

A
b ← BIT COUNT OF MAP
IF MOST OF THE BITS IN MAP ARE ON OR OFF, SET TO ZERO MAP AND DEVIATION
b < TMAP OR b >16-TMAP ?
YES
dY ← dCr ← dCb ← MAP ← 0
B
NO
REDUCE mY TO BYUNI BITS
REDUCE mCr TO BCUNI BITS
REDUCE mCb TO BCUNI BITS
REDUCE dY TO BYPAT BITS
REDUCE dCr TO BCPAT BITS
REDUCE dCb TO BCPAT BITS
MAP = 0 ?
NO
FIND NEW MAP USING MAP, TMAP, MAPGRP SAVING NEW MAP, MAP BIT CODE
(FIG. 15)
YES
BIT COUNT OF MAP TO NEW MAP CHANGED ?
NO
YES
ADJUST dY, dCr, dCb BY
√(16-BIT COUNT (NEW MAP)) / BIT COUNT (NEW MAP)
√(16-BIT COUNT (MAP)) / BIT COUNT (MAP)
B

FIG. 14B

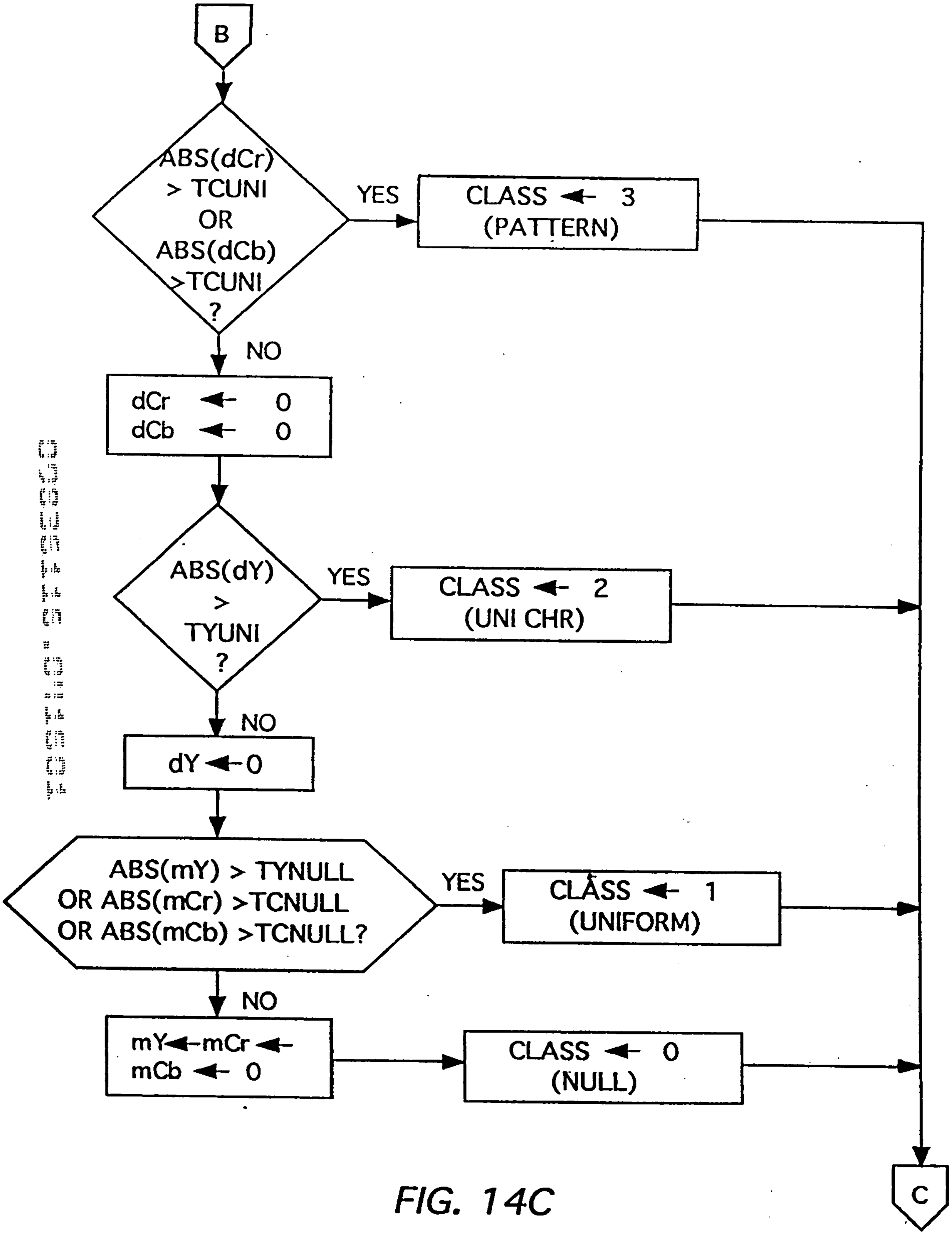

B
ABS(dCr)
> TCUNI
OR
ABS(dCb)
>TCUNI
?
YES
CLASS ← 3
(PATTERN)
NO
dCr ← 0
dCb ← 0
ABS(dY)
>
TYUNI
?
YES
CLASS ← 2
(UNI CHR)
NO
dY ← 0
ABS(mY) > TYNULL
OR ABS(mCr) >TCNULL
OR ABS(mCb) >TCNULL?
YES
CLASS ← 1
(UNIFORM)
NO
mY ← mCr ←
mCb ← 0
CLASS ← 0
(NULL)
C

FIG. 14C

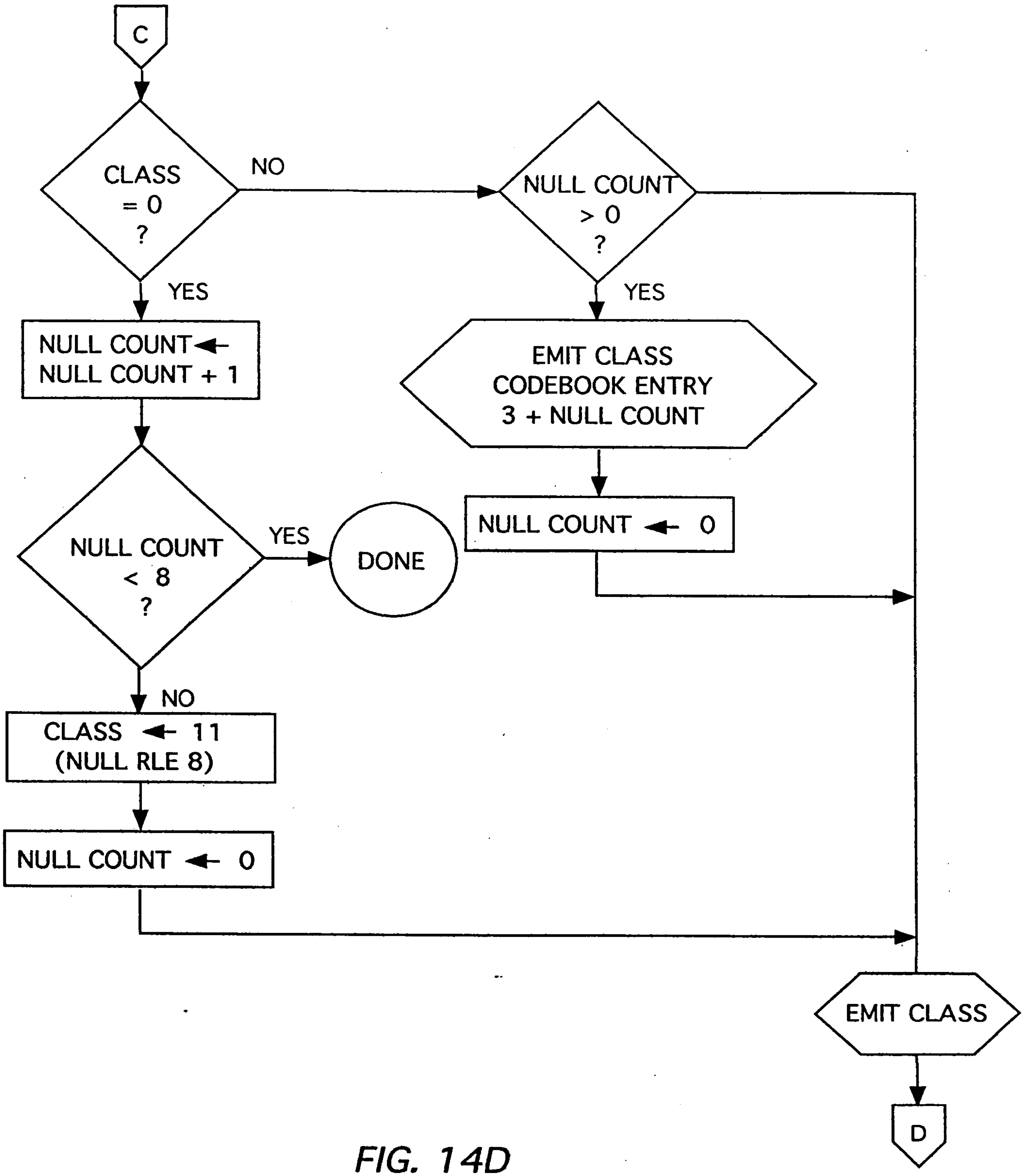

C
CLASS = 0 ?
NO
YES
NULL COUNT > 0 ?
YES
NULL COUNT ← NULL COUNT + 1
EMIT CLASS CODEBOOK ENTRY 3 + NULL COUNT
NULL COUNT ← 0
NULL COUNT < 8 ?
YES
DONE
NO
CLASS ← 11 (NULL RLE 8)
NULL COUNT ← 0
EMIT CLASS
D

FIG. 14D

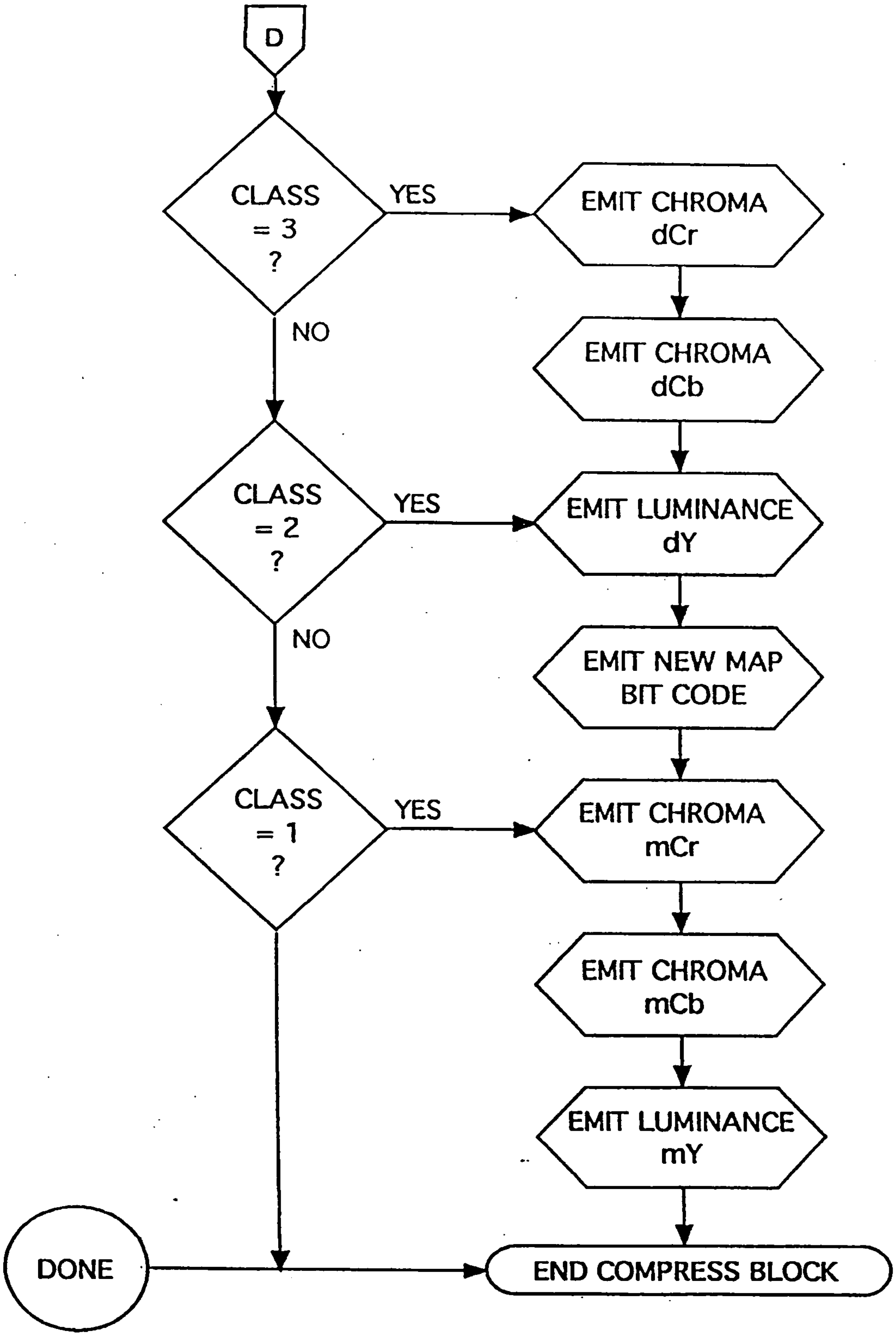

D
CLASS = 3 ?
YES
NO
EMIT CHROMA dCr
EMIT CHROMA dCb
CLASS = 2 ?
YES
NO
EMIT LUMINANCE dY
EMIT NEW MAP BIT CODE
CLASS = 1 ?
YES
EMIT CHROMA mCr
EMIT CHROMA mCb
EMIT LUMINANCE mY
DONE
END COMPRESS BLOCK

FIG. 14E

FIG. 15A

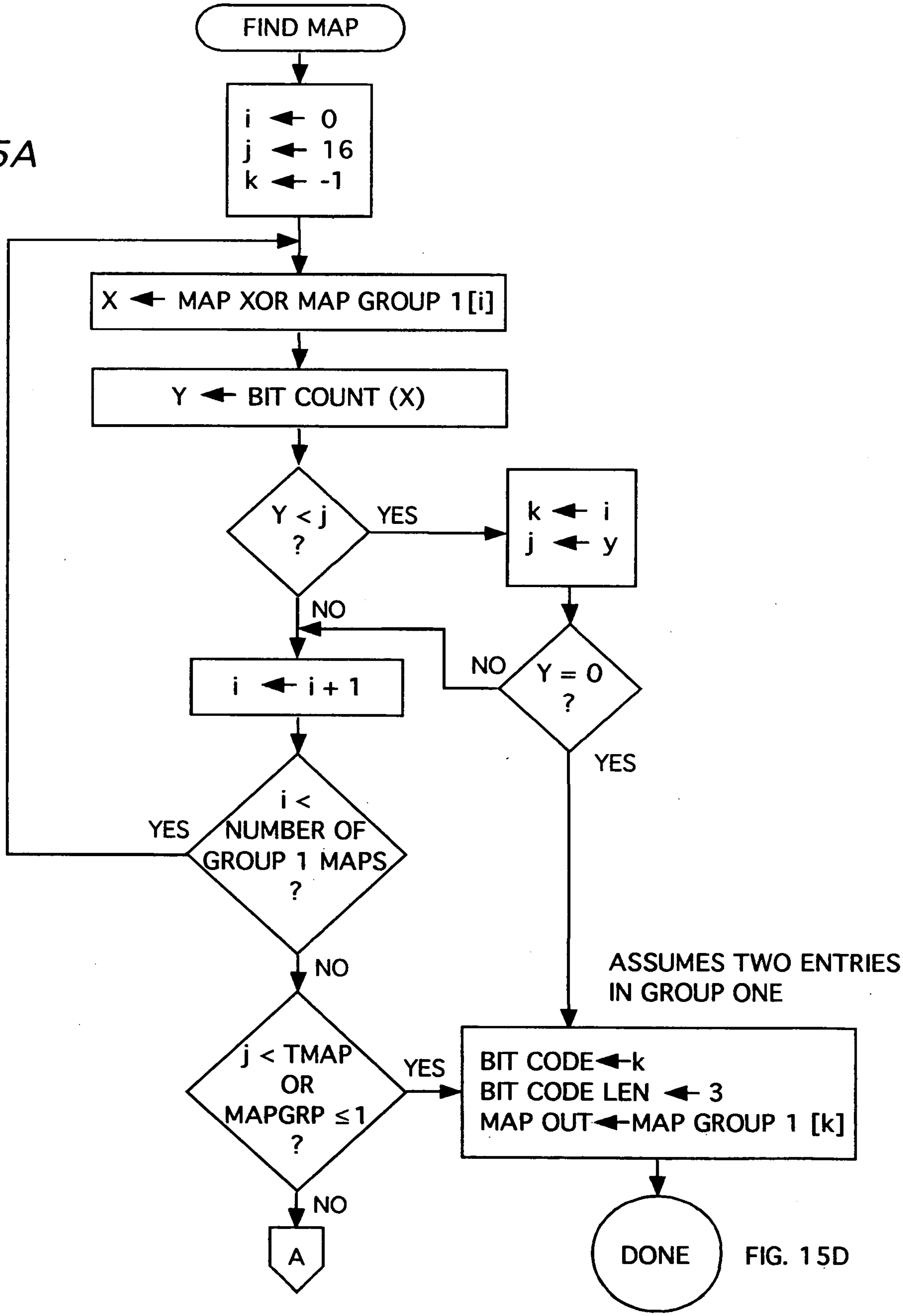


FIG. 15D

FIG. 15B

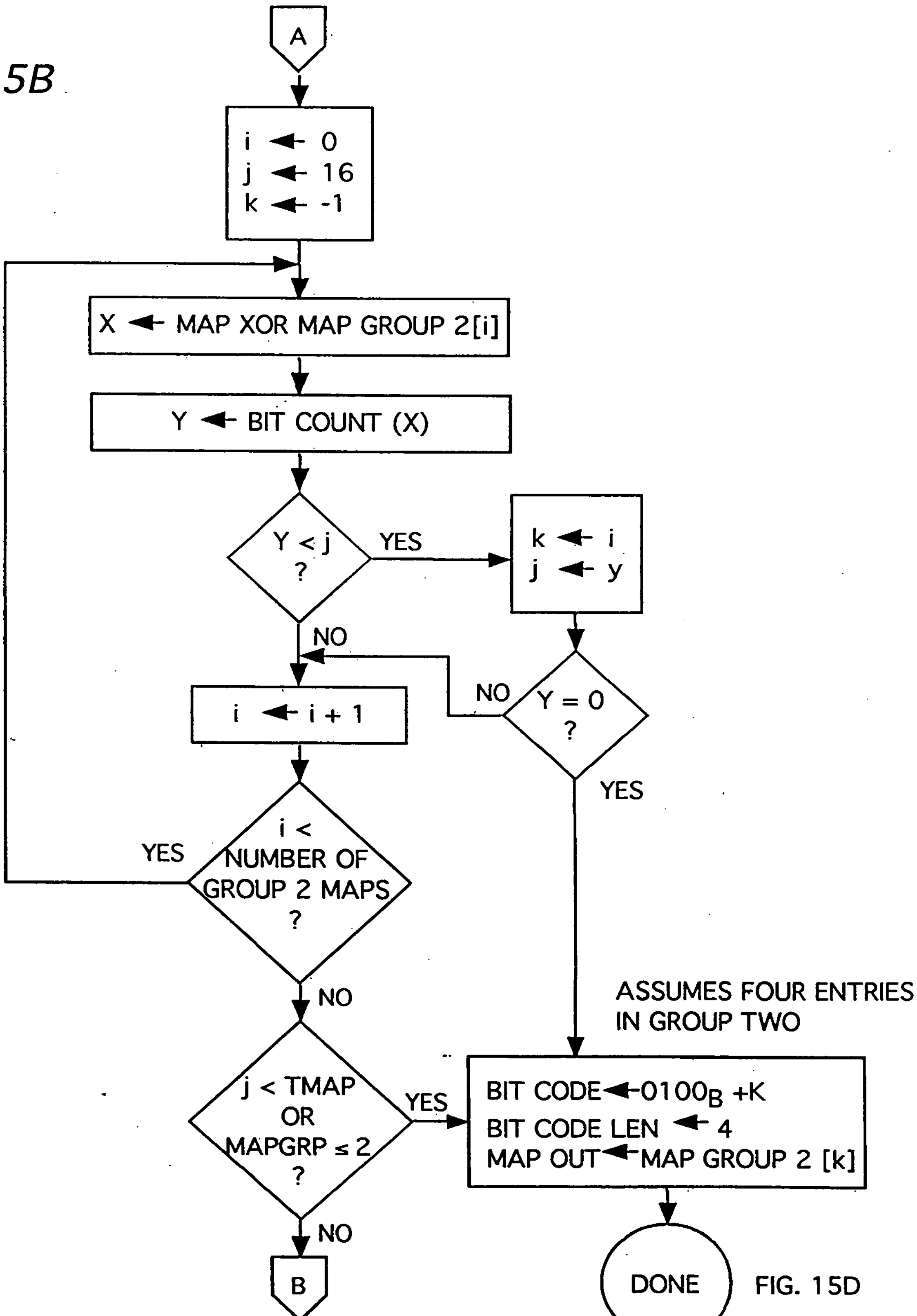

A
i ← 0
j ← 16
k ← -1
X ← MAP XOR MAP GROUP 2[i]
Y ← BIT COUNT (X)
Y < j ?
YES
k ← i
j ← y
NO
Y = 0 ?
NO
YES
i ← i + 1
i < NUMBER OF GROUP 2 MAPS ?
YES
NO
ASSUMES FOUR ENTRIES IN GROUP TWO
j < TMAP OR MAPGRP ≤ 2 ?
YES
BIT CODE ← 0100B +K
BIT CODE LEN ← 4
MAP OUT ← MAP GROUP 2 [k]
NO
B
DONE
FIG. 15D

FIG. 15C

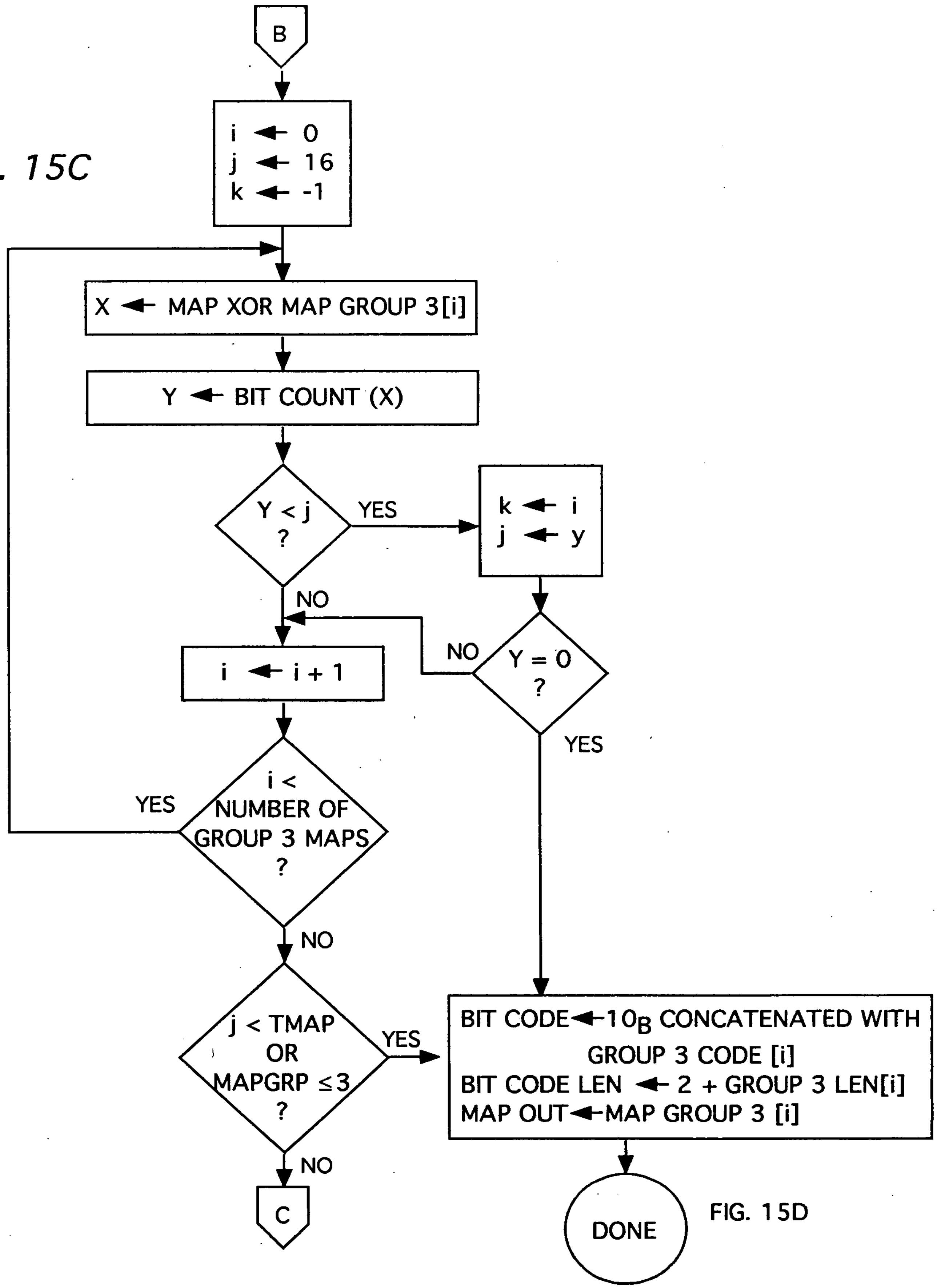


FIG. 15D

D	D	D	D	D	D
D	C	B	B	C	D
D	B	A	A	B	D
D	B	A	A	B	D
D	C	B	B	C	D
D	D	D	D	D	D

FIG. 16A

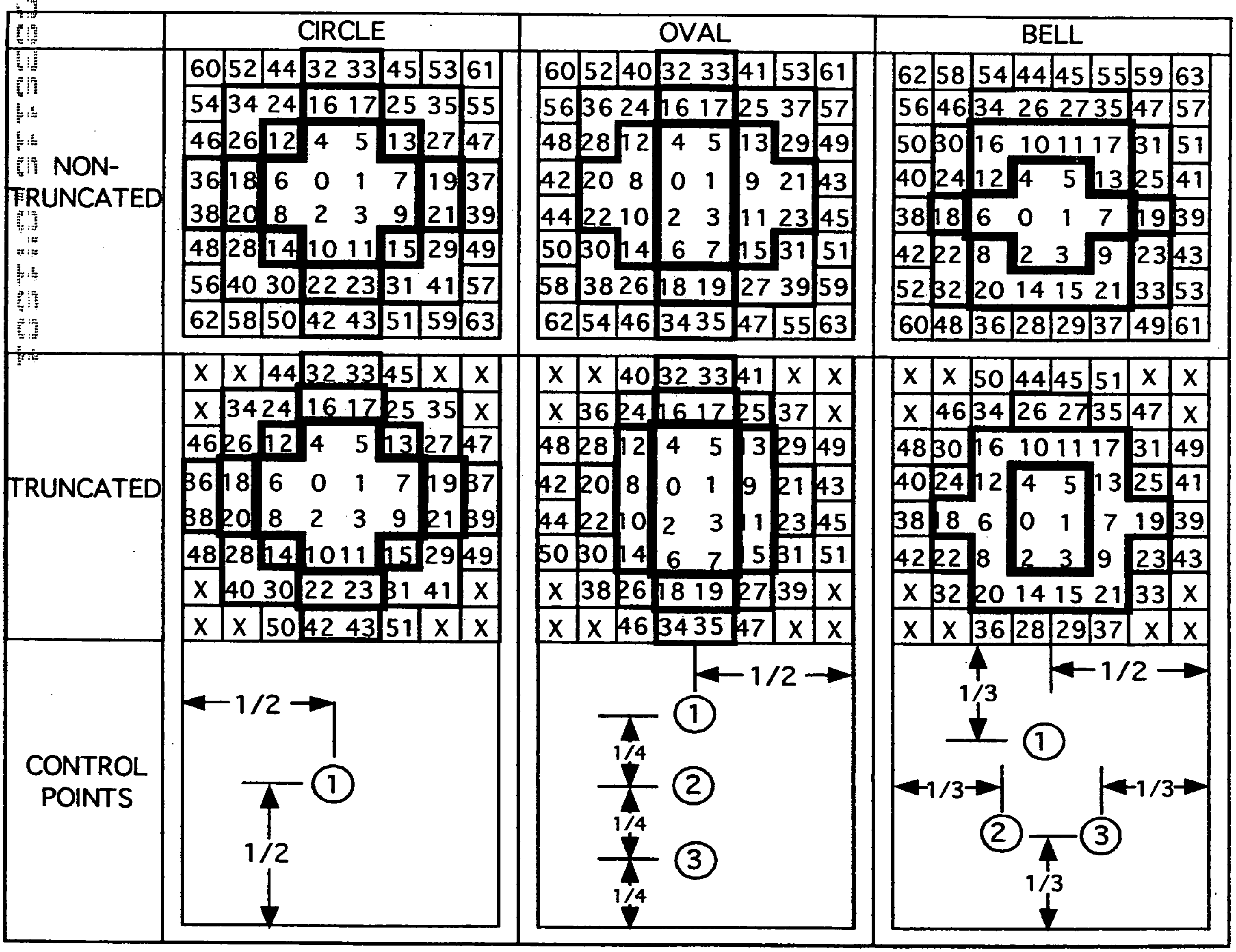


FIG. 16B

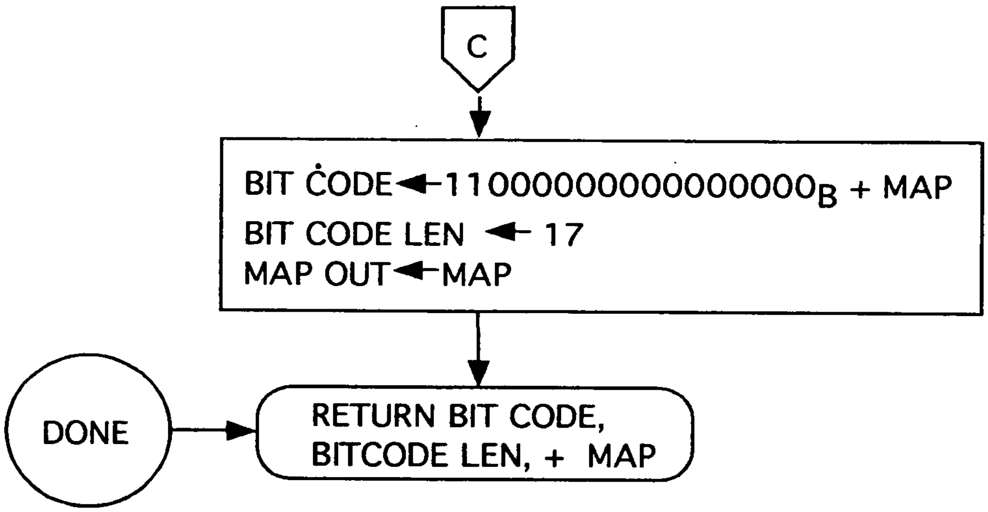


FIG. 15D

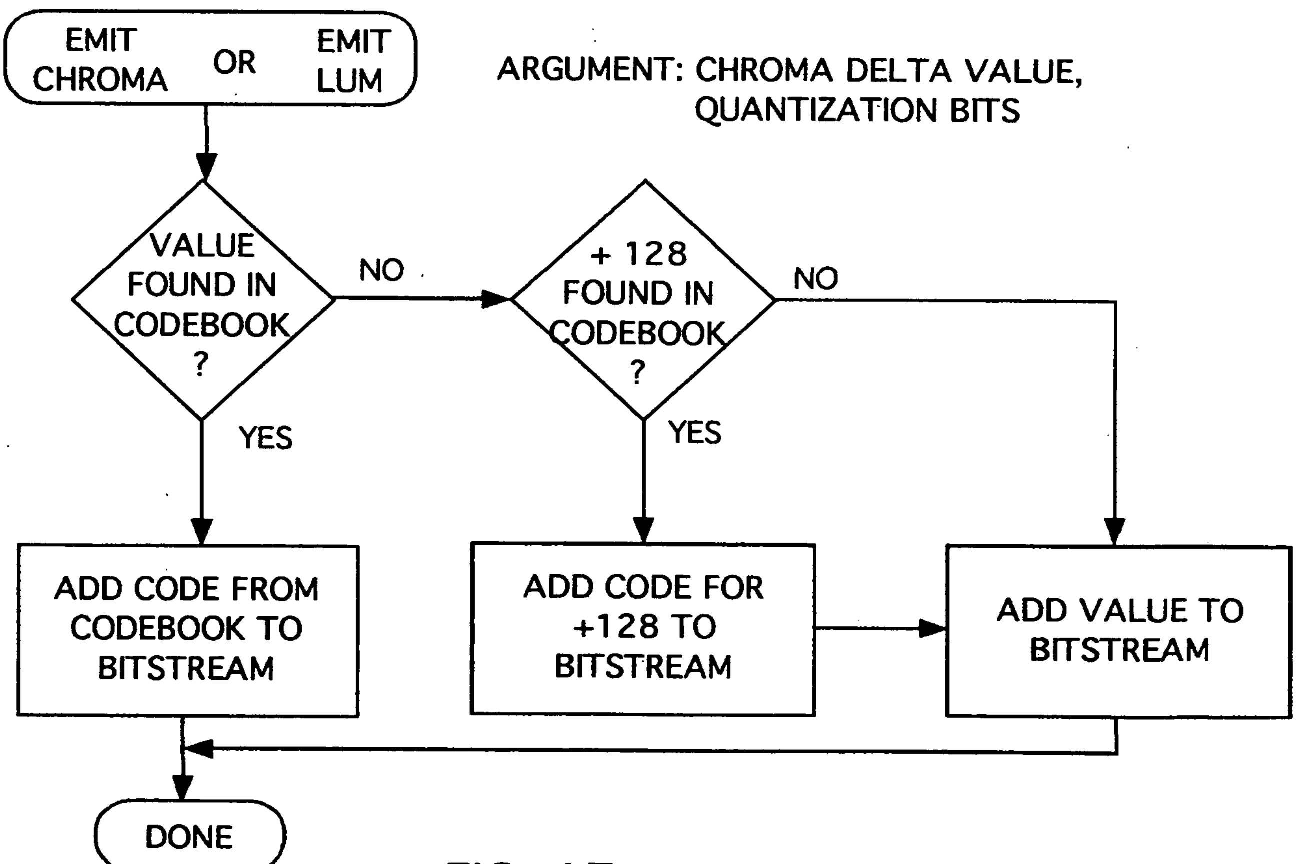


FIG. 17

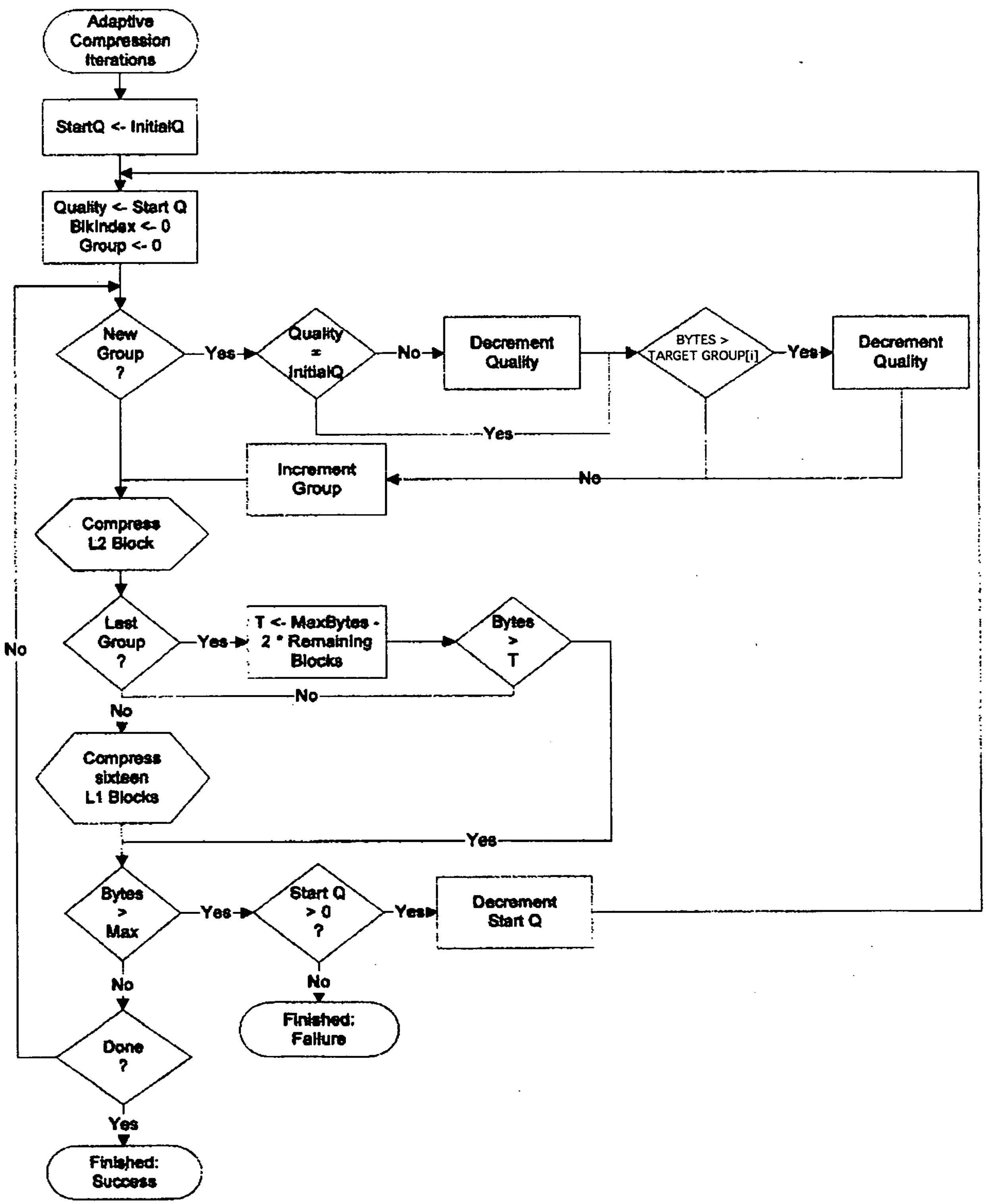

Adaptive Compression Iterations
StartQ <- InitialQ
Quality <- Start Q
BlkIndex <- 0
Group <- 0
New Group ?
Yes
Quality = InitialQ
No
Decrement Quality
BYTES > TARGET GROUP[i]
Yes
Decrement Quality
Yes
No
Increment Group
Compress L2 Block
Last Group ?
Yes
T <- MaxBytes - 2 * Remaining Blocks
Bytes > T
No
No
Compress sixteen L1 Blocks
Yes
Bytes > Max
Yes
Start Q > 0 ?
Yes
Decrement Start Q
No
Finished: Failure
No
Done ?
No
Yes
Finished: Success

FIG. 18

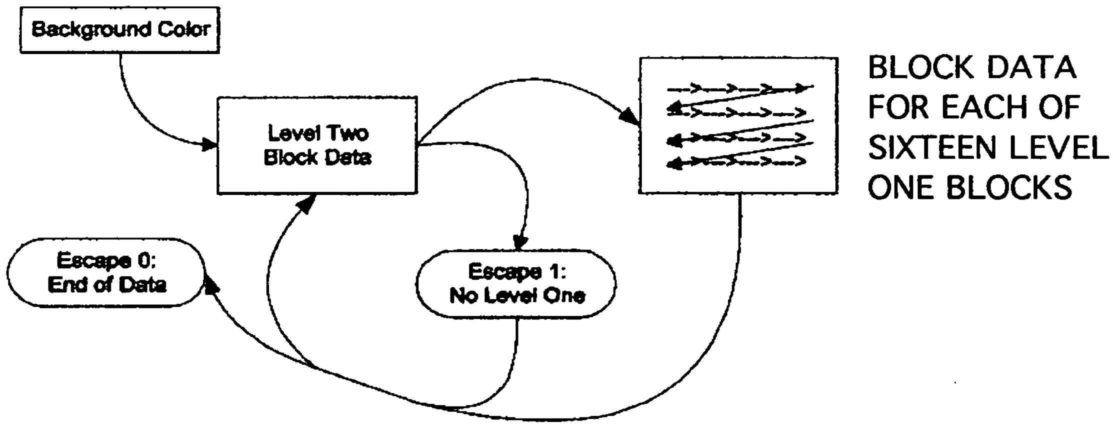


FIG. 19

Input				
A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	Q	R	S	T
U	V	W	X	Y

FIG. 20A

Light Edge Filter				
A	3B/4 C/4	3C/4 B/4	D	E
3F/4 K/4	9G/16 3H/16 3L/16 M/16	9H/16 3G/16 3M/16 L/16	3I/4 N/4	3J/4 O/4
3K/4 G/4	9L/16 3M/16 3G/16 H/16	9M/16 3L/16 3H/16 G/16	3N/4 I/4	3O/4 J/4
P	3Q/4 R/4	3R/4 Q/4	S	T
U	3V/4 W/4	3W/4 V/4	X	Y

FIG. 20B

Medium Edge Filter				
A	2B/3 C/3	2C/3 B/3	D	E
2F/3 K/3	4G/9 2H/9 2L/9 M/9	4H/9 2G/9 2M/9 L/9	2I/3 N/3	2J/3 O/3
2K/3 F/3	4L/9 2M/9 2G/9 H/9	4M/9 2L/9 2H/9 G/9	2N/3 I/3	2O/3 J/3
P	2Q/3 R/3	2R/3 Q/3	S	T
U	2V/3 W/3	2W/3 V/3	X	Y

FIG. 20C

Heavy Edge Filter				
A	B/2 C/2	C/2 B/2	D	E
F/2 K/2	H/4 G/4 M/4 L/4	H/4 G/4 M/4 L/4	I/2 N/2	J/2 U/2
K/2 F/2	H/4 G/4 M/4 L/4	H/4 G/4 M/4 L/4	N/2 I/2	O/2 J/2
P	Q/2 R/2	R/2 Q/2	S	T
U	V/2 W/2	WW V/2	X	Y

FIG. 20D

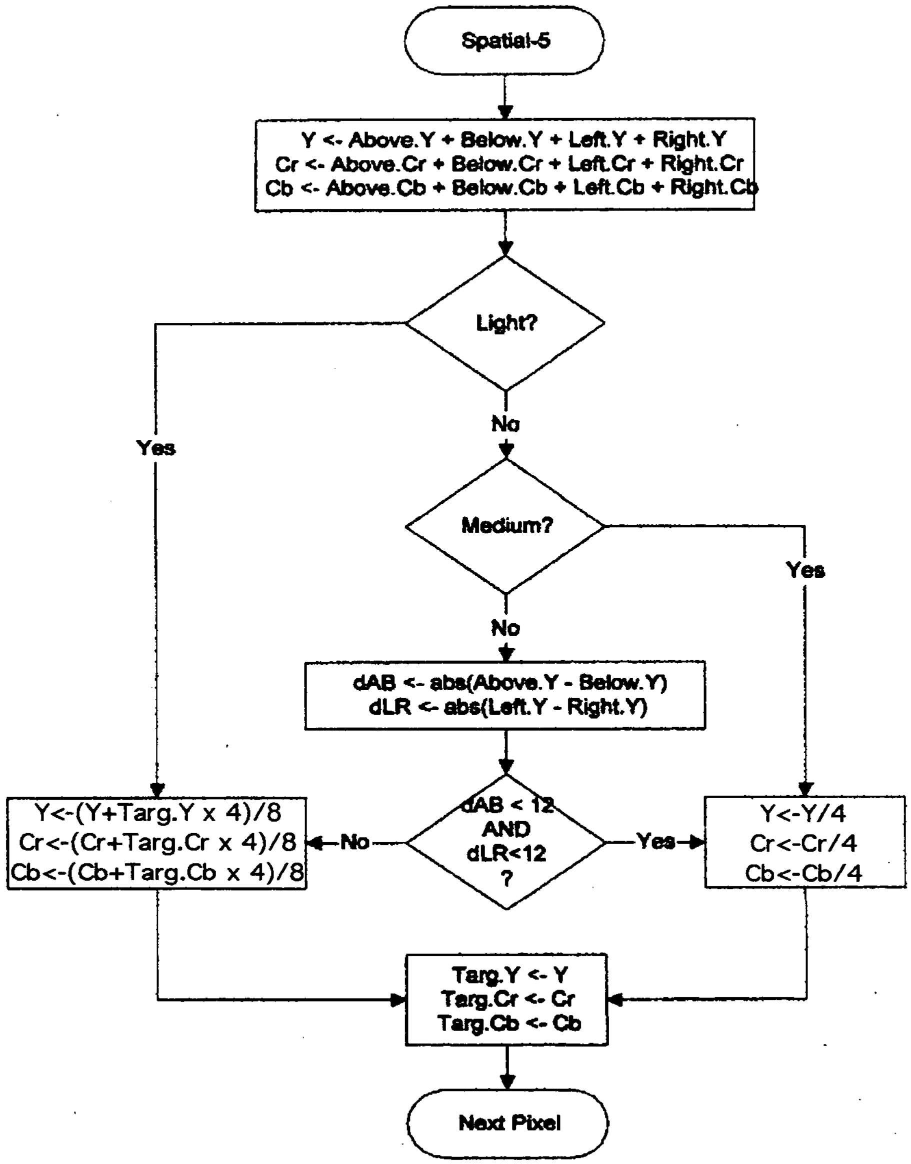

Spatial-5
Y <- Above.Y + Below.Y + Left.Y + Right.Y
Cr <- Above.Cr + Below.Cr + Left.Cr + Right.Cr
Cb <- Above.Cb + Below.Cb + Left.Cb + Right.Cb
Light?
Yes
No
Medium?
Yes
No
dAB <- abs(Above.Y - Below.Y)
dLR <- abs(Left.Y - Right.Y)
dAB < 12 AND dLR<12 ?
No
Yes
Y<-(Y+Targ.Y x 4)/8
Cr<-(Cr+Targ.Cr x 4)/8
Cb<-(Cb+Targ.Cb x 4)/8
Y<-Y/4
Cr<-Cr/4
Cb<-Cb/4
Targ.Y <- Y
Targ.Cr <- Cr
Targ.Cb <- Cb
Next Pixel

FIG. 21

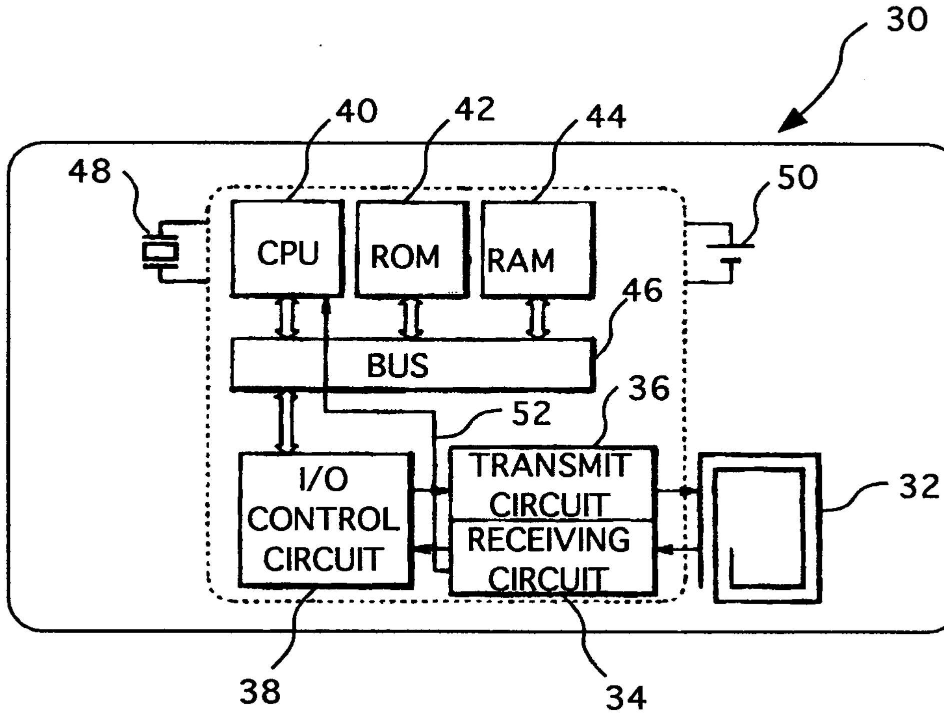

30
40
42
44
48
50
CPU
ROM
RAM
46
BUS
36
52
I/O
CONTROL
CIRCUIT
TRANSMIT
CIRCUIT
RECEIVING
CIRCUIT
32
38
34

FIG. 22